LAB #1: INTRODUCTION TO CRYPTOGRAPHY
DUE DATE: February 10, beginning of the lecture

TO BE DONE IN GROUPS OF TWO

1. General Description

The notion of cryptography consists of hiding secret information from non trusted peers by mangling messages into unintelligible text that only trusted peers can rearrange. In this lab, we will use and compare three different techniques commonly employed to hide or encrypt information: secret key cryptography (DES), public key cryptography (RSA) and message digests (SHA-1).

There is a group of built in functions that will help you complete this assignment. These functions are included in Openssl. Openssl is an open source toolkit that implements security protocols such as SSL but also includes a general purpose cryptography library which you will use. Openssl is usually part of most of the recent linux distributions.

2. File with sample codes and results

We are giving a group of files (compressed in tar.gz) as a reference on how to use certain built in functions that we will mention in this handout and that you must use in your implementations. These files are just skeletons that you should modify in order to obtain the expected results. Included in the compressed file is also a test case for your DES-CBC implementation (test.txt and test.des) and a file with general descriptions of some built in functions. Please download the file lab1.tar.gz from http://shay.ecn.purdue.edu/~ece495f/labs/lab1/lab1.tar.gz

3. DES encryption / decryption

In this part of the lab, we will be coding a tool to encrypt and decrypt files using DES in mode CBC (Cipher Block Chaining). “tempdes.c” is a skeleton file that encrypts/decrypts a fixed 64-bit block. In this assignment, you will extend this skeleton code to take an arbitrarily sized input file and encrypt/decrypt it, by implementing the Cipher Block Chaining DES mode of operation. You must actually implement the CBC mode, and you are not allowed to use any built-in function besides what is present in tempdes.c. You can find information about DES-CBC in your text book.

You may want to check your work against the input file "test.txt". If you have implemented the algorithm correctly, you should get the output in "test.des". These files are part of lab1.tar.gz.

3.1 Requirements

a. Just use the built in functions that appear in tempdes.c
b. Your code should result in an executable of the following form:
   ./tempdes iv key inputfile outputfile

   The parameters description is as follows:
   - iv: the actual IV to use: this must be represented as a string comprised only of hexadecimal digits.

- key: the actual key to use: this must be represented as a string comprised only of hexadecimal digits.
- inputfile: input file name
- outputfile: output file name

For example:
./tempdes fecdba9876543210 0123456789abcdef test.txt test.des

If any of the arguments is invalid, your code should return an appropriate message to the user. Be sure to consider the case when the keys are invalid.

4. Performance measures for DES, RSA and SHA1

The final part of this lab consist on measuring the time DES, RSA and SHA-1 take to process files of different sizes.

a. Generate text files with the following sizes:
   - For DES (in bytes): 8, 64, 512, 4096, 32768, 262144, 2047152
   - For SHA-1 (in bytes): 8, 64, 512, 4096, 32768, 262144, 2047152
   - For RSA (in bytes): 2, 4, 8, 16, 32, 64, 128

b. Encrypt and decrypt all these files using the DES function that you wrote in part 3. Measure the time it takes to encrypt and decrypt each of the files. To do this, you might want to use the C function "gettimeofday". Add these timing functions to your implementation of part 3.

c. Measure the time for RSA encryption and decryption for the file sizes listed in part a. To do this, make appropriate changes to the file "temprsa.c". This skeleton code shows how to use built-in RSA encryption and decryption functions, but you will need to augment it to allow for reading from arbitrary files, and to insert necessary instrumentation code for timing purposes.

d. Measure the time for SHA-1 hash generation for the file sizes listed in part a. To do this, make appropriate changes to the file "tempsha1.c". This skeleton code shows how to use built-in SHA-1 hashing functions, but you will need to augment it to allow for reading from arbitrary files, and to insert necessary instrumentation code for timing purposes.

e. Prepare a report of your observations in the format requested under "**Deliverables" (section 6).**

5. Non-Programming Problems:

a. Suppose the DES mangler function (or function ***F)*** mapped every 32 bit value to zero, regardless of the value of its input. How does the output after Round 16 of DES compare to the input at the start of Round 1 of DES? Show your work and provide appropriate explanations.

b. Consider the following message digest (hash) function: Take the input messages, divide it into 128-bit chunks. XOR all the chunks together to get a 128-bit result. Run a standard message digest algorithm on the result. Is this a good message digest function?

c. Perform encryption and decryption using the RSA algorithm for the following: p = 3, q = 11, e =7, M = 5. Please show all the steps in a systematic fashion, like in page 270 of the Stallings text book.
d. In the public-key system using RSA, you intercept the cipher text C=10 sent to a user whose public key is *e*=5, *n*=35. What is the plaintext M? Show the steps.

6. Deliverables:

a. A physical copy of the file tempdes.c, submitted using the "turnin" command. Name your file <login1>.<login2>.des.c, where login1 and login2 are login names of the two members of the group. For example: sanjay.rtorresg.des.c
- Submission instructions
   - *Log into shay.ecn.purdue.edu*
   - *Move to the same directory as the file you are going to submit.*
   - *Execute: "turnin –c ece495f –p lab1 <login1>.<login2>.des.c"*

b. Printout.
Turn in a **typed** report with the following information:
   - C code implemented in parts 3, 4.c and 4.d
   - Graphs showing: (i) DES encryption / decryption times; (ii) RSA encryption times; (iii) RSA decryption times; and (iv) SHA-1 digests generation times. In each of these graphs, the X axis should plot the file sizes in units of bytes, and the Y axis should plot time measurements in units of microseconds (μs).
   - Answer the following questions:
     - Compare DES encryption and RSA encryption. Explain your observations.
     - Compare DES encryption and SHA-1 digest generation. Explain your observations.
     - Compare RSA encryption and decryption times. Can you explain your observations? [Hint: Refer to section 6.3.4.3 of the book "Network Security, private communication in a public world" by Kaufman, Perlman and Speciner.]
   - Answers to all non-programming problems in Part 5.

7. Note on Grading

Points will be deducted if any of the deliverables in Section 6 is missing.
Further, points will be deducted if the program implemented in section 3
- Does not have a name as specified in section 6.a, (or)
- Does not compile (or)
- Does not produce the expected results. Please make sure you compare your results against the given test case.